Detection of Crawler Traps: Formalization and Implementation Defeating Protection on Internet and on the TOR Network

Maxence Delong[1], Baptiste David[1] and Eric Filiol[2,3]
[1]*Laboratoire de Virologie et de Cryptologie Opérationnelles, ESIEA, Laval, France*
[2]*Department of Computing, ENSIBS, Vannes, France*
[3]*High School of Economics, Moscow, Federation of Russia*

Keywords: Crawler Trap, Information Distance, Bots, TOR Network.

Abstract: In the domain of web security, websites want to prevent themselves from data gathering performed by automatic programs called bots. In that way, crawler traps are an efficient brake against this kind of programs. By creating similar pages or random content dynamically, crawler traps give fake information to the bot and resulting by wasting time and resources. Nowadays, there is no available bots able to detect the presence of a crawler trap. Our aim was to find a generic solution to escape any type of crawler trap. Since the random generation is potentially endless, the only way to perform crawler trap detection is on the fly. Using machine learning, it is possible to compute the comparison between datasets of webpages extracted from regular websites from those generated by crawler traps. Since machine learning requires to use distances, we designed our system using information theory. We used wild used distances compared to a new one designed to take into account heterogeneous data. Indeed, two pages does not have necessary the same words and it is operationally impossible to know all possible words by advance. To solve our problematic, our new distance compares two webpages and the results showed that our distance is more accurate than other tested distances. By extension, we can say that our distance has a much larger potential range than just crawler traps detection. This opens many new possibilities in the scope of data classification and data mining.

1 INTRODUCTION

For most people, surfing the web is done via a web browser where it is just about to click on a few links. But the ecosystem is not only made of humans. Indeed, there are automatic programs, called bots, that allow you to automatically browse websites. Browsing motivations of the latter are multiple. Indeed, they can be used for archiving, search engine optimization, content retrieval, user information collection — with or without agreement —search for non-posted and potentially confidential documents, intelligence gathering for attack preparation... Usually, bots are efficient and largely faster than humans to collect all information. If the motivations of such tools are more or less legitimate, it is understandable that some sites do not wish to be analyzed in such a way.

To counter these unwanted automatic analysts, websites have implemented different strategies. From the detection of bots with basic methods, to challenges offered that only humans (Turing test) can solve (or supposed to be), various methods have emerged across the last years. In a few cases, the most advanced ones have the ability to neutralize the importune bot. The main technique used to deceive and hinder bots' operations is generically described as "Crawler traps". As always in IT security, what refers to detection measures and protection on one side, also refers to bypass techniques symmetrically. Hence the aim of our paper is to formalize the concept of crawler trap detection and management, to present different techniques capable of interfacing with anti-bot countermeasures and to present our detection results.

More than trying to be stealthy (it would result in reducing the bot's performance and therefore losing its main interest), we want to counter the measures put in place to neutralize bots. The objective is to have ways to pass where the other widely used bots generally stop and fail, trapped by the website's defenses. Our whole problem is to have the operational means to be able to detect whenever a website traps our bot and therefore to be able to avoid it. Surprisingly, there are neither no research no real public operational solutions on the subject, which means we are going to

Delong, M., David, B. and Filiol, E.
Detection of Crawler Traps: Formalization and Implementation Defeating Protection on Internet and on the TOR Network.
DOI: 10.5220/0009367207750783
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 775-783
ISBN: 978-989-758-399-5; ISSN: 2184-4356

explore a new field.

This paper is therefore organized to address this problem. Section 2 presents the state of the art on the different website protection mechanisms. Section 3 addresses te formalization we have considered and presents the solutions implemented based on the theoretical aspects we have considered. Section 4 presents the results we have obtained from our solutions during a large number of experiments both on the Internet and on the TOR network (aka Darkweb). Finally, Section 5 resumes our approach and possible uses.

2 STATE OF ART

As far as web bots are concerned, there is a great diversity of uses. If the majority of bots as chatbots (Rouse, 2019) or Search Engine Optimization (SEO) bots as Googlebot (Google, 2019) are very helpful for websites owners, some of them are malicious. The purpose of malicious bots can be multiple. Among many example, we can mention spambots which collect email addresses to prepare phishing campaigns. Subsequently to fight against botnets' activities, many security measures have been developed.

Most solutions degrade the user experience or come to be impractical from a user point of view. An example of a repressive solution is the IP banishment. If the web server guessed that a bot is crawling it, an error code can be returned on legitimate webpages thus disallowing content access. In the case where the website has incorrectly guessed and real user has been blocked, the frustration is high. Another example of common impractical security on the web are captchas (Hasan, 2016). Indeed, if originally, most of captcha were used to validate specific operation where a human user was mandatory (online payment, registration...), nowadays, websites more and more condition the access to the total content of the website by resolving a captcha first. But this is not a sufficient solution since there exists practical ways to automatically bypass them (Sivakorn et al., 2016). As a consequence, other security systems have been developed to be more efficient.

Usually, the *official* way to prevent web-crawlers is to create a file named "*robot.txt*" put at the root of the website. In this file, all the section forbidden for crawling are referenced to prevent legitimate bots from falling into them. This file is only declarative and there is no deeper protection to forbid crawling. In a way, it is a gentleman's agreement that determined bots shamelessly bypass. Moreover, the "*robot.txt*" give direct information to the attacker on sensitive parts of the website.

The new trend to provide real protection is focused on systems called "*spider trap*" or "*crawler trap*". The goal is to ambush the malicious bot on a dedicated trap inside the website. The intent is to imprison the bot in a specific place where it is supposed continue to operate endless false information (or the same one, again and again) in order to finally waste time and resources. There are many ways to create and deploy a crawler trap.

Involuntarily Crawler Traps. Surprisingly, the notion of crawler trap exists from a long time, since some of them were not designed to be a crawler trap. This is what we call *involuntarily crawler trap*. For instance, for management purpose, we can find online interactive calendar which are constituted by an infinite number of page, one for each month, generated on the fly by the webserver. Nowadays, most of calendars are developed in javascript but this language is prohibited on the TOR network for security reasons. This is why we can still find such objects online. Another example stands in some website which generate pages on the fly from a general link but with different parameters. Incorrectly managed by the bot, it is possible to make them endlessly loop on this type of link.

Infinite Depth or Infinite Loop. In contrast, we can find intentional crawler which are designed for that purpose. The most used solutions stands in the automatically generated pages when the system detects a bot. The content generated does not matter as long as the crawler hits the trap and that it crawls fake sub-pages. In the majority of cases, a loop with relative URL is created and the bot will crawl the same set of two pages. For instance, we can use two twin directories referencing each other such as the bots comes to one to go to the other and vice-versa:

```
http://www.example.com/foo/bar/foo/
    bar/foo/bar/somepage.php
```

Identifiers. For some website, the strategy is to generate a unique identifier store in the page URL. This one is assigned to each client visiting it in order to track him. This one is usually called a *session ID*. The bot — as any user —will receive such identifier. The trap lies in the fact that the user ID is changing randomly at random time. Such a way, the bot can be redirected to already visited webpages but with a different ID — an transparent action a regular user will not perform. The session ID is usually implemented in the URL as follows:

```
http://example.com/page?sessionid=3
    B95930229709341E9D8D7C24510E383
http://example.com/page?sessionid=
    D27E522CBFBFE72457F5479117E3B7D0
```

There exists countermeasures to bypass such protection for bots, starting by registering hashes of already visited webpages to a smarter implementation of URL management.

Random Texts. A different solution, which — in addition to its deployment —is more effective than others, stands in random generation of text in a webpage. With simple script from webserver side, one page can be randomly generated with random text and random links referencing the same page. Once this page is hit, the bot will treat what looks to it as *different* webpage, since it comes from different links with a different page's name, over and over.

But whatever is the level of sophistication of the crawler trap, it can only generate pages that respect the strategies for which it has been designed. It means that each crawler trap has specific characteristics. Either the web page is created automatically on the fly and the content is fully random, or it starts from an existing webpage whose content is randomly derived. In other words, the crawler can be working on totally fake platforms, where the pages are similar to the original site in terms of shape, HTML architecture, even some key words, part of content...

This is this last category of crawler trap which matters for us. Indeed, the involuntary one can be avoided by a bot aware of such objects. And about intentional ones, where they are based on URL management, it is possible to bypass the security with a specific management of already visited pages. But the last technique with random text generation cannot be managed in the same way and it requires a specific approach to handle it. This is the purpose of our paper.

From an operational point of view, we are visiting websites, page after page. The main difficulty lies in the fact that the trap generates random pages after the bot has visited a certain number of pages (based on the speed of the visit, the number of visits, the logic of the order of the links visited, etc). In a way, we can only base our decisions on the already visited pages and the ones we are going to visit next. This is the operational context in which we operate. Our whole approach is not to avoid being detected by the website (crawling performance would be too downgraded) but to detect whenever the website starts setting a trap. For this reason, we need to measure the distance between regular and irregular webpages.

3 USE OF DISTANCES

The objective is to measure the distance between several data sets to perform a distinction between several families. This can be done in section 3.1 in order to define precisely what we are really trying to measure and the approach which drives our forthcoming analyses. Subsequently, we will be able to present some of the existing distances in the section 3.2 and then we will discuss the contributions proposed by our distance to answer our problematic.

3.1 Approach to Resolve the Problem

From a generic point of view, we have two *ethnic groups*: pages extracted from regular website and pages generated by crawler traps. In each of the ethnic group, it is possible to find different *families*. In the case of regular websites, it means that all the pages of a family come from the same website. The same applies for pages generated from different crawler traps.

In our case, we are trying to detect whenever there is a webpage generated from a crawler trap in a set of regular webpages. According to our operational context, we are aggregating webpages on the fly, which means we cannot know the full dataset in advance. It means we have to check the difference of a new webpage from a set of already visited webpages. This is doable by computing a distance $\mathcal{D}$ between the new webpage and the current set. In that sense, we can assume that a family F composed of n samples, $\forall i, j \in \mathbb{N}^{+*}$ such as $1 \leq i \leq j \leq n$ we have the random value X which models the distance between two samples s_i and s_j such as $X = \mathcal{D}(s_i, s_j)$. Since all samples belong to the same family, there is no reason that the distance between two samples is different to a third one, that is to say $\forall i, j, k \in \mathbb{N}^{+*}$ where $i \neq j \neq k$ we have $\mathcal{D}(s_i, s_j) \approx \mathcal{D}(s_i, s_k)$.

For optimization purposes, we cannot compute a distance between the new one and all pages of the current set. Since the distance between webpages of a single family is supposed to be the same, we can consider the mean (expected value) as a good estimator. This mean is computed on the fly by updating it with each new value which belong to the family. It means that our detector system is based on the fact that the distance between a new sample s and the mean of a family m is based on the fact that $\mathcal{D}(s_i, s_j) \leq \varepsilon$ where ε is distinctive for each each family.

The challenge is therefore to define a distance $\mathcal{D}$ that respects the property defined bellow. It means that our distance must be *discriminating* in the sense that each family must have its own mean of distance. And it should be *accurate* which means that the standard deviation from the mean is supposed to be as small as possible.

The reduction of the mean makes sense in terms of confidence intervals. Indeed, when applied to our case, such intervals are composed by $\left[\bar{x} - t_\alpha \frac{s}{\sqrt{n}}; \bar{x} + t_\alpha \frac{s}{\sqrt{n}}\right]$ where $\bar{x}$ is the mean of the family, t_α is the confident level selected, s the standard deviation observed and n the size of the current family. The goal is to avoid the overlapping of several confidence intervals and the best way to achieve such a goal when we cannot increase the size of the family (since it could introduce crawler-trapped pages) is to reduce the standard deviation.

3.2 Distances Evaluated

As part of our research, we studied many distances. Among those selected, we kept two of them with the requested properties as described in section 3.1: the Jensen-Shannon distance (JSD) and the Hellinger distance (HD). Moreover we have designed a custom distance to solve our problem more efficiently.

For all distances, we consider the content of two webpages as a random value X and Y composed of words. Each word (or group of words) in a page has a probability of occurrence. The set of all of these probabilities constitutes a probability distribution usually noted P and Q from the same space $\mathcal{X}$.

3.2.1 Jensen-Shannon

The *Jensen-Shannon divergence* (JSD) (Lin, 1991) is based on the idea that we should be able to balance the different distributions involved in the divergence, according to their importance. As for previous distributions, it aims at measuring the diversity (or the dissimilarity) between two populations. With $\pi_1, \pi_2 \geq 0$ real positive numbers such as $\pi_1 + \pi_2 = 1$ which are the *weights* of the two probability distributions, we define the Jensen-Shannon divergence as:

$$JS_\pi(P,Q) = H(\pi_1 P + \pi_2 Q) - \pi_1 H(P) - \pi_2 H(Q)$$

where H is the entropy function as defined by Shannon (Shannon, 1948) on a random variable such as:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

In order to manipulate a true metric, it is common to compute the square root of the JSD divergence such as:

$$D_{JS_\pi}(p_1, p_2) = \sqrt{JS_\pi(p_1, p_2)}$$

3.2.2 Hellinger

The *Hellinger distance* (although known under the name of *Matusita measure* (Matusita, 1955)) has been designed to check if one probability distribution follows another probability distribution and if it is a true metric (Comaniciu et al., 2003). The normalized version (the values are between 0 and 1) is as follows (Nikulin, 2011):

$$\mathcal{D}_H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2}$$

3.2.3 Custom Distance - Shark

The main drawback of the existing distances we observed is that the random variables they consider are defined on the same space $\mathcal{X}$. It makes sense in simple cases but our one is a bit more specific. Indeed, we are computing the set of words page after page and it means we do not have access to the set of all words. In a page, we can find words which are not a present in the next page (and vice-versa) for the sake of example.

The main issue when we use existing distance is that, to be mathematically exact, we must remove unique words from each page. The possibility to preload all possible words from website is out of scope in our operational context since we compute the set of words, page after page, on the fly and considering unknown words on a page with a probability of zero does not fulfill all mathematical designs (we artificially increase the number of words and we change the probability distributions to remain consistent). Such a loss deprives us of information and therefore a loss of accuracy for our measurements. The idea of our measure has its roots in this simple observation.

The idea of one of our measures, called *Shark* (others will be provided in an extended version) is based on the rate of the entropy of the symmetric difference on the entropy from the two distributions. In other words, we look at the information contribution of the exclusive words from the two pages on all the information embedded in these two pages. Considering $P \in \mathcal{X}$ and $Q \in \mathcal{Y}$ we have $P \setminus Q = \{\forall x \in X \text{ where } x \notin Y\}$ where we define our measure such as:

$$\mathcal{D}(P,Q) = \frac{H(P \setminus Q) + H(Q \setminus P)}{H(P) + H(Q)}$$

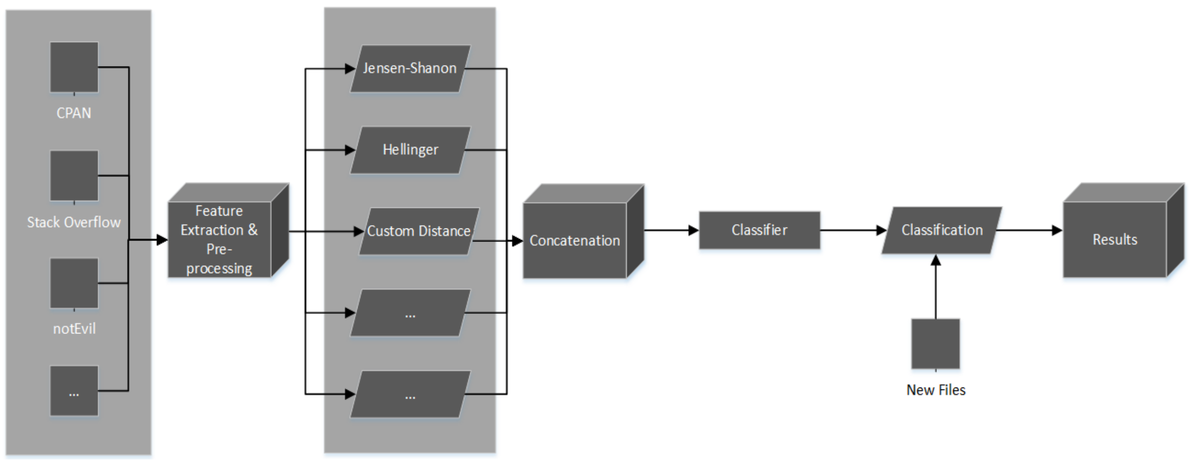

Figure 1: Full process to determine a page family.

4 EXPERIMENTS

When we have conducted these experiments, we did not forget the main objective of detecting a crawler trap automatically. From an operational point of view, the crawler processes a website, page after page, and can fall into a crawler trap at any time. It is nearly impossible to detect a crawler trap with only two consecutive pages. The concept we developed is to train a classifier progressively with n consecutive pages then to detect whenever a page is too different or too similar from the previous ones.The process is schematized in Figure 1. At the time, only the creation of a more discriminating measureis presented. The implementation or the choice of a classification algorithm and the operational tests will be the outcome of future works or an extended version.

Since we have defined properties and the protocol to evaluate different distances, it is relevant to focus on which data we are going to process. First, we propose to explain which are the websites used to help our procedure to distinguish between regular websites (for what it could mean) from crawler traps. Then, we propose to explain which data are relevant in a web page so that it will help to build efficient detection system.

4.1 Experimental Dataset

The approach we propose is requiring the presence of a dataset, which will be presented in the section 4.1.1. More than explaining the context of our paper and the different actors involved, the idea is also to open our approach to make it reproducible and leave the results being critically analyzed, by anyone, as we did in section 4.2.

4.1.1 Websites

As there is no dataset available for our problematic — it has not been studied in the past —, we created our own. In this dataset, we chose to collect webpages from 13 distinct sources, each with different linguistic properties. To gather the dataset, we used a simplified version of our crawler. For some webpages like Wikipedia, we ”clicked” on the random generation of article on the left side of the page (available in all languages) to have various panel of subjects. The inescapable component is that all pages are one click away (from a bot point of view) from the previous and next one, hence they all are consecutive pages.

In the collected set, 4 sources are from a crawler trap and 9 are from ”classic” websites. We collected 500 pages from each sources. There are divided in two main categories: regular websites and crawler trap ones. The first is about ”*classic*” websites that anyone can easily find online. We wanted to have enough material from ”*spoken-language*” to have relevant tests. Among the different websites evaluated for this purpose, we have selected the following detailed list:

- **CPAN - 1 Set (F1):** ”*Comprehensible Perl Archive Network*” is a documentation for each Perl libraries. It can be a very long explanation or just few lines of codes. People wrote the documentation in a good English.
- **Jeuxvideo.com - 1 Set (F2):** *JeuxVideo.com* is a well known French website for news and discussions about video games. A part of the website is a forum for teenagers with very few moderation. The language used is an approximate French, not

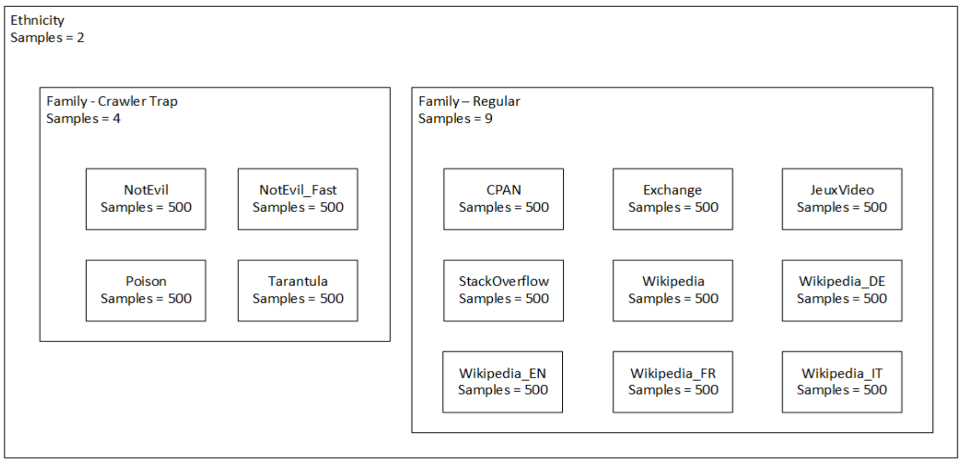

Figure 2: Summary of data.

well mastered but still understandable for native speakers.

- **Stack Exchange - 1 Set (F3):** Network composed of 174 communities. The topics covered are various and range from science to religion through history, literature or mathematics. A proper English is mandatory for the validation of the question asked.
- **Stack Overflow - 1 Set (F4):** One of the network in *Stack Exchange* specialized in computer programming questions. There is less English sentences and more sources codes as an example of their issues.
- **Wikipedia - 5 Sets:** *Wikipedia* is the most known free encyclopedia. The language is very accurate. We have downloaded 5 sets, one by language (German (F5), English (F6), French (F7) and Italian (F8)) and the last one is composed of a combination of this four previous languages (F9).

The second category is about crawler-trap websites. We have to admit that it exists really few crawler-traps, at least identified. Why? Because most of the techniques used are based on simple values saved in cookies in web browsers. They are not hard to bypass for a trained enough bot. The real difficulties arise when the targeted website adapts the content of the webpage when it is close to detect a bot instead of a real human. To achieve such a goal, it exists few scripts or programs able to perform this task. More than a long list, the following one aims to represent all the diversity we have found about:

- **notEvil - 2 Sets:** *notEvil* is originally a search engine in the TOR network. On this website, there is an embedded game named Zork. This game is associated with an identifier and, on the page, there are three links to a new game. Hence, a crawler trap is created because once the bot hits this link, it will play games indefinitely. On the same page, a small part is for "Recent Public Channels" and changes every time a channel is created or someone responds to an old one. To determine if this change has an impact on your classification, we collected two sets: one is composed of webpages downloaded (F10) as the bot does (everything is collected in the same minute) and one set (F11) is for a long time gathering process (one webpage per hour). This changing part is usually spambots, URL links or unethical topics.
- **Poison - 1 Set (F12):** *Poison* (poi,) is the most random crawler trap we ever met. There is no practical example available online, therefore we have downloaded the software (developed in 2003) and generated our sample of 500 pages. Through this, we had the possibility to play with several coefficients to generate a real random sample of pages:
  - Number of paragraphs [1:10]
  - A CSS background (adding HTML tags and data to create a colorful page) [0:1]
  - Minimum number of words per paragraph [25:75]
  - Number of words added to the minimum number of words per paragraph [25:100]

  This crawler trap is based on the operating system dictionary, thus, a lot of words are uncommon and very advanced, even for native speakers (ex:*thionthiolic*). In this system, the probability of picking up the word *"house"* is the same as the word *"chorioretinitis"*.
- **Tarantula - 1 Set (F13):** This is the name of a

PHP crawler trap available on GitHub (tar,). It generates a simple page with 1 to 10 new links and a text that contains between 100 and 999 words chosen randomly in a list of 1000 predefined words. *Tarantula* is not online so we have configured it on our local web server and generated 250 samples from the root address and 250 by following a random link of the page.

In order to ensure *reproducibility* (Nosek et al., 2015) of results presented here, the dataset (514.3 MiB uncompressed) will be made available on the third author's webpage.

4.1.2 Relevant Data in a Webpage

There is no defined rules for a functioning crawler trap. It can be located on a small part of the webpage just as it can be on the whole page. And, as explained previously in section 1, a web page is composed of different parts. A crawler trap can have an impact on all of these different parts. The shape of the page (more precisely the way or the style in which it is written) and the content (words which are displayed to user's eyes) are equally relevant to classify. Therefore, we divided the page in three parts and created four main features:

– A list of words contains in the <*body*> tag of the page. To collect them, we used the following regex: m/\w+/. We convert all words in lower case to normalized data.

– The HTML tag architecture starting at the <*body*> tag. To remove all text between each tag, we used the regex s/>[ˆ<]+</></. We also removed comments and links found in <*src*> and <*href*> tags.

– The metadata of the page located in the <*head*>. Comments and text between tags were removed.

– For comparison purposes, the fourth category is composed of the list of words and the HTML architecture. The same pre-processing were applied for each part.

If the content and the HTML tag architecture seems to be the most relevant data to exploit, some crawler traps are not smart enough to change the header for example. In that case, the metadata are signing and distinct features. We did not add coefficient between each feature to keep them with an equal weight.

4.2 Experimental Results

Since the content of the material used for experiment has been described, it matters to evaluate our measure comparing to existing ones. Empirical results that we have been able to evaluate, using words split one by one does not necessarily give very precise results. Indeed, words are often used in a given context, which means that they are linked each others. It is for this reason that we decided to evaluate the probability of clusters of consecutive words. Our best results are in the case where we are using bi-grams with overlapping extraction. Provided results are given in the following tables, where columns represent each a family as described in section 4.1 and rows the means and the associated standard deviation for the given distances.

The first point to note is that each distance does not give the same values, even for the same family. In truth, what matters is that each family, for a given distance, has more or less a unique value to be enough accurate. It is the standard deviation that allows us to discuss accuracy. The results clearly show that, on average, the standard deviation of the proposed measure is smaller than that of other distances.

Indeed, in the case of Jensen-Shannon distance, when we are looking at the raw date from the page (the most relevant information we can extract here), values of distance are usually located around the average value of 0.25 with an average standard deviation of 0.10. This is almost the same situation for Hellinger distance which has, on the same type of data, all its distances close to an average of 0.18 with an average standard deviation of 0.08. The situation differs with Shark. Indeed, the average distance is 0.95 and the standard deviation is about 0.03 with a range of values larger than others. This is the difference of range between vales from the group of regular web pages and those coming from crawler trap worlds which is relevant to perform the discrimination between the two groups thanks to a small standard deviation. This consequence avoids the phenomenon of overlapping between families..

One interesting point lies in the ability for Shark measure to produce bigger values than one. This is due to the way we decide to design of the probabilities in the set resulting of the symmetrical difference. Indeed, those ones can be build intrinsically which means probabilities are computed internally in each set resulting of the difference or extrinsically which means probabilities are computed considering all the different sets (exclusive ones and the common one). Going beyond unity is a rare phenomenon that can be observed on large sets where the ratio of common and different words is enough strong (which is the case on crawler traps) when using probabilities designed in an intrinsically way.

Considering the average of the distances as the sum of normalized random variables, it becomes pos-

JSD	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
Content	0.30	0.13	0.18	0.26	0.15	0.16	0.19	0.18	0.16	0.00	0.12	0.22	0.19
	0.16	0.09	0.07	0.08	0.09	0.09	0.09	0.09	0.10	0.00	0.03	0.14	0.06
HTML	0.00	0.00	0.00	0.00	0.41	0.52	0.48	0.47	0.48	0.00	0.03	0.25	0.00
	0.00	0.00	0.00	0.00	0.14	0.15	0.14	0.14	0.15	0.00	0.02	0.13	0.00
Header	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Raw	0.26	0.13	0.18	0.26	0.31	0.42	0.38	0.38	0.41	0.00	0.08	0.28	0.19
	0.15	0.09	0.07	0.08	0.14	0.16	0.14	0.14	0.16	0.00	0.02	0.11	0.06

Hellinger	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
Content	0.22	0.09	0.13	0.19	0.11	0.12	0.14	0.13	0.12	0.00	0.09	0.16	0.89
	0.12	0.07	0.05	0.06	0.06	0.07	0.07	0.06	0.08	0.00	0.02	0.10	0.28
HTML	0.00	0.00	0.00	0.00	0.30	0.38	0.35	0.35	0.35	0.00	0.02	0.18	0.00
	0.00	0.00	0.00	0.00	0.10	0.12	0.11	0.11	0.12	0.00	0.02	0.10	0.00
Header	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Raw	0.19	0.09	0.13	0.19	0.23	0.31	0.28	0.28	0.30	0.00	0.05	0.20	0.13
	0.11	0.06	0.05	0.06	0.11	0.12	0.11	0.11	0.12	0.00	0.02	0.08	0.04

Shark	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
Content	0.94	0.84	0.92	0.93	0.94	0.94	0.94	0.95	0.97	0.37	0.98	1.00	0.00
	0.04	0.05	0.03	0.03	0.02	0.02	0.03	0.02	0.02	0.03	0.09	0.01	0.00
HTML	0.71	0.00	0.00	0.00	0.96	1.00	0.99	0.98	0.98	0.74	0.73	0.56	0.00
	0.15	0.00	0.00	0.00	0.05	0.07	0.05	0.06	0.05	0.00	0.01	0.19	0.00
Header	0.75	0.48	0.85	0.72	0.66	0.66	0.65	0.68	0.79	0.00	0.00	0.71	0.00
	0.02	0.02	0.03	0.02	0.03	0.04	0.04	0.02	0.08	0.00	0.00	0.09	0.00
Raw	0.94	0.83	0.92	0.93	0.95	0.97	0.96	0.97	0.98	0.62	1.23	1.02	1.00
	0.04	0.05	0.03	0.03	0.03	0.04	0.03	0.03	0.03	0.02	0.11	0.02	0.00

Figure 3: Detection Results for JSD, Hellinger and Shark distances.

sible to apply the consequences of the central limit theorem which says that the resulting variable (average of distances, in our case) tends towards the normal distribution. Thus, the interval $[\bar{x}-\sigma;\ \bar{x}+\sigma]$ should contain 68 % of the observed values of measures inside elements from given family and the interval $[\bar{x}-3\sigma;\ \bar{x}+3\sigma]$ should hold 99 % .

For this reason, a small standard deviation allows us to have a more efficient distance than those evaluated here. This is the main idea of our evaluation of different measures we have used to perform an efficient detection. More details will be provided in an extended version.

5 CONCLUSION

In our study, we have defined a new measure to compare two objects (from a family to a single sample) and we have applied it to the case of crawler trap detection. We can decide with a reasonable accuracy if a web page belongs to a given family (crawler trap or not) with better results compared to existing information distances. Our results are characterized with a lower standard deviation which mean less overlapping between families.

Since we can detect that we are about to be trapped by the targeted website, the ability to avoid it can be computed not so hardly. For the sake of simplicity about the procedure to use , it would require to check a subset of previously visited websites and to avoid to go on pages with a distance value too far from the one actually measured in the current set. Consequently, we can focus on relevant links and avoid trapped ones.

Of course, just talking about distance efficiency is not enough to claim the accuracy of our results. But we wanted to present the way we designed measure and procedure we develop to solve this problem which does not already exist in academic world. In an extended version, we will present our system coupled with statistical tests able to say if a new sample can be integrated to an existent family (since its distance with it is enough small) or not. The most important part has been covered with confidence intervals on which tests are based. The notion of overlap rates from confidence intervals is central to assessing and explaining the effectiveness of our distances from existing ones. This one needs to be further developed in future work.

In addition with the extended version of the current paper , we would present more distances and more details about how to use them in the case of detection. Extended results will be provided to definitively prove efficiency of methods we are using. This future work would aim to complete the theoretical background which stands behind our concepts of entropy based on symmetrical difference from different sets.

Of course, this new concept of distance can be applied to different fields of datamining and anywhere machine learning is relevant, as Hellinger or Jensen-Shannon distances are usually used . If the goal is still to provide better and more efficient ways to breakthrough against crawler traps in order to allow bots to escape them, the area of exploitation can be expanded to different fields. Actually, it should be relevant to any problem based on a dataset crafted on the fly (or which cannot be known in advance) or where manipulated data are heterogeneous in their characteristics. From a general point of view, applications and further developments can be infinitely vast.

REFERENCES

Stickano/tarantula: Spider trap. https://github.com/Stickano/Tarantula. Accessed: 2019-11-21.

Sugarplum – spam poison. http://www.devin.com/sugarplum/. Accessed: 2019-11-21.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577.

Google (2019). Googlebot definition.

Hasan, W. (2016). A survey of current research on captcha. *International Journal of Computer Science & Engineering Survey*, 7:1–21.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Trans. Information Theory*, 37:145–151.

Matusita, K. (1955). Decision rules, based on the distance, for problems of fit, two samples, and estimation. *The Annals of Mathematical Statistics*, 26.

Nikulin, M. (2011). Hellinger distance. *Encyclopedia of Mathematics*.

Nosek, B., Alter, G., Banks, G., Borsboom, D., Bowman, S., Breckler, S., Buck, S., Chambers, C., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D., Hesse, B., Humphreys, M., and Yarkoni, T. (2015). Promoting an open research culture. *Science (New York, N.Y.)*, 348:1422–5.

Rouse, M. (2019). Chatbot definition.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Sivakorn, S., Polakis, J., and Keromytis, A. D. (2016). I ' m not a human : Breaking the google recaptcha.